Using Brutus - A Brief Walkthrough

Arturo Moncada Torres

April 22, 2012
Preface

HPC Cluster Brutus (or simply Brutus) is ETHZ’s high performance computer cluster. It can be very helpful when you need to perform really heavy simulations, algorithms, or calculations. Using it is very easy, but you need to know a few tricks. This walkthrough pretends to help a new user getting started with Brutus and do some basic stuff with it. Although the guide is oriented towards a Windows user, I believe Mac and Linux fans will also find it useful.

Brutus is being improved and updated all the time. The information here corresponds to that available at the moment of writing.
## Contents

1 Introduction ................................................................................. 3
   1.1 What is Brutus? ................................................................. 3
   1.2 How does a cluster (Brutus) work? ................................. 3
   1.3 Who can use Brutus? ......................................................... 4
   1.4 Brutus' technical specifications ....................................... 4

2 Getting ready ................................................................................ 6
   2.1 Getting a Brutus account .................................................. 6
      2.1.1 As an ETHZ member .................................................. 6
      2.1.2 As an external collaborator ....................................... 7
   2.2 Getting an SSH client to login to Brutus ......................... 8
   2.3 Getting an FTP client to manage files in Brutus ............... 9
   2.4 Important Considerations ............................................... 10

3 Using Brutus ............................................................................... 12
   3.1 Basics ................................................................................. 12
   3.2 Modules ............................................................................. 13
   3.3 Submitting a (batch) job .................................................. 15
   3.4 Outputs ............................................................................. 18
   3.5 Logging out ....................................................................... 18

4 Troubleshooting .......................................................................... 19
   4.1 Important Considerations Using Different Software in Brutus 19
      4.1.1 MATLAB ................................................................. 19

5 Closing comments ....................................................................... 20
   5.1 Acknowledgements .......................................................... 20
   5.2 Using this walkthrough .................................................... 20

Bibliography .................................................................................... 21
Chapter 1

Introduction

First of all, we need to know what we are dealing with.

1.1 What is Brutus?

Brutus is ETHZ’s central High Performance Computer (HPC) Cluster. Actually, “Brutus” stands for Better Reliability and Usability Thanks to Unified System. A cluster is a collection of individual computers (nodes), each one with its own processor(s), memory and storage, connected together via a common network. HPC clusters clusters typically contain hundreds or thousands of nodes, connected together (via a high-speed network such as InfiniBand).

1.2 How does a cluster (Brutus) work?

All the participating systems in a cluster have the same operating system and software running. A user interactively connects to a login node, edits his programs and input data for the calculations and sends it to a batch system which manages the available resources and shares them between all users in a fair manner. The batch system is completely transparent to the user so that he does not have to take care where his jobs are running. To the user the whole cluster appears as a single system.

For optimal benefit of a cluster, tasks should be scheduled in parallel. An application can run on a single core or on multiple cores on the same computer node (in Brutus’ current version, this occurs in between 2 and 16 cores). Another part of applications may request a higher number of cores. These run on multiple compute nodes and have to communicate for synchronisation. This can be done with standard communication libraries as the Message Passing Interface (MPI).
1.3 Who can use Brutus?

Brutus is open to all interested and qualified members of ETHZ. Researchers from other Swiss and international institutions can use it too, as long as they have a collaboration with an institute of ETHZ.

Brutus is jointly owned by nearly 50 professors—the so called shareholders—in 12 departments and the IT Services, who are responsible for the acquisition and management of the system. Each professor (shareholder) is guaranteed a share of CPU time proportional to his/her investment. The part financed by the IT Services is made available to the whole scientific community of ETHZ. Shareholders who buy into a part of the cluster are guaranteed partial use of the cluster corresponding to their share while all other users share the available resources among themselves.

1.4 Brutus’ technical specifications

Brutus is a heterogeneous collection of different hardware generations. It is continuously expanded and updated with the latest hardware technology available on the market. The following information corresponds to that available at the moment of writing. You will find updated information in Brutus Wikipedia page.

Brutus contains 12 different kinds of compute nodes:

- **Regular nodes**
  - 104 nodes with four 16-core AMD Opteron 6174 CPUs and 64 GB of RAM (6656 cores)
  - 24 nodes with two 12-core AMD Opteron 6174 CPUs and 32 GB of RAM (576 cores)
  - 410 nodes with four quad-core AMD Opteron 8380 CPUs and 32 GB of RAM (6560 cores)
  - 80 nodes with four quad-core AMD Opteron 8384 CPUs and 32 GB of RAM (1280 cores)
  - 256 nodes with two dual-core AMD Opteron 2220 CPUs and 16 GB of RAM (1024 cores)

- **Large-memory (fat) nodes**
  - 64 nodes with four 16-core AMD Opteron 6174 CPUs and 256 GB of RAM (4096 cores)
- 6 nodes with two 12-core AMD Opteron 6174 CPUs and 128 GB of RAM (144 cores)
- 10 nodes with four quad-core AMD Opteron 8380 CPUs and 128 GB of RAM (160 cores)
- 7 nodes with eight dual-core AMD Opteron 8220 CPUs and 64 GB of RAM (112 cores)

• GPU nodes
  - 18 nodes with two 12-core AMD Opteron 6174 CPUs, 32 GB of RAM and 2 Nvidia Fermi C2050 CPUs (432 cores + 36 GPUs)
  - 2 nodes with two 6-core AMD Opteron 2435 CPUs, 32 GB of RAM and 6 Nvidia Tesla C1060 GPUs (24 cores + 12 GPUs)
  - 2 nodes with two 6-core AMD Opteron 2435 CPUs, 32 GB of RAM and 6 Nvidia and AMD GPUs (24 cores + 2 CPUs)

All the nodes are connected to the cluster’s Gigabit Ethernet backbone. All the nodes (except those with Opteron 2220 CPUs) are connected to a high-speed InfiniBand QDR network.

Thanks to its heterogeneous nature, Brutus can tackle a wide range of applications, for example:

• Serial and embarrassingly parallel computations
• Distributed-memory computations (Open MPI, Quadrics MPI)
• Shared-memory, multithreaded applications (openMP) up to 128 GB of memory and 16 threads
• Third-party (commercial) application such as Ansys CFX, Ansys FLUENT, MATLAB, etc.

In total Brutus contains over 18,400 cores plus a few hundreds in the cluster’s file servers, login nodes, management nodes, etc. The peak performance of Brutus is approximately 120 teraflops (120 \times 10^{12} floating-point operations per second).
Chapter 2

Getting ready

In order to be ready to use Brutus, you will need the following:

1. A Brutus account.
2. A third-party secure shell (SSH) client to connect to Brutus.
3. A file-transfer protocol (FTP) client to manage your files in Brutus.

Don’t worry, getting these is easier than it sounds. Besides, we will go through each of them step by step.

2.1 Getting a Brutus account

You will need an account to log in and use Brutus. Depending on your status, you need to follow a different procedure.

2.1.1 As an ETHZ member

If you are a member of ETHZ with a valid NETHZ account, you need to simply fill out the Brutus account request form. You will need to provide your NETHZ username and password to confirm your identity and access this form. The following information is required to fill in the form:

* **Username** The Brutus username is by definition your NETHZ username. It cannot be changed.

* **Name** The full name is taken from NETHZ. It can be modified if needed.
* E-mail Taken from NETHZ. It can be modified if needed.

• Phone number A day-time phone number is highly desirable.

* Shareholder group Use the drop-down menu to indicate if you belong to a shareholder group. If this is not the case or your are unsure, leave it to “NONE” and write a comment in the appropriate box. Select “STUDENT” only if you are attending a parallel computing class. Remember student accounts are only valid for the duration of the class (usually one semester).

* Supervisor Should be a professor or equivalent.

* Address of institute Provide the full address of the institute.

* Project description Describe in 2 or 3 sentences the kind of jobs you intend to run and their resource requirements (such as number of processors, duration, memory and disk space, etc.). It does not need to be very specific, a general overview is enough.

• Comments Fill it if you have any special needs or to ask a question.

The fields marked with “*” are mandatory.

2.1.2 As an external collaborator

If you are not a member of ETHZ, you may request an account as long as you have a collaboration with an institute of ETHZ. The procedure in this case is slightly different.

You will need to ask your contact at ETH – your “sponsor” – to create a NETHZ guest account for you. If your sponsor does not know how to do that, he/she should contact the IT Support Group of his/her department or the NETHZ administration. The creation of a NETHZ guest account may take a few days. Once you have received your NETHZ account, simply follow the procedure for ETH members as described above. You will have to indicate the name of your sponsor in the “Supervisor” field.

In any case, account requests are normally processed within one working day. You will be notified by e-mail when your account is ready. If you have not received an answer after two days, you will need to contact Cluster Support (cluster-support@id.ethz.ch).
2.2 Getting an SSH client to login to Brutus

You will need to connect to Brutus using your NETHZ username and the corresponding password. For this, you will need a third-party secure shell (SSH) client to connect to Brutus. Personally, I recommend PuTTY, but there are many others (like Exceed).

Go to PuTTY’s download page and select the version that best fits you (by “you” I mean your computer). Once the download is finished, move the file to the location you prefer (e.g. Program files, Desktop, My Documents, etc.).

PuTTY requires no installation, since the file you downloaded is the executable itself you will be using. Double click it and you will open the PuTTY’s configuration screen (Fig. 2.1). Just enter the parameters as shown in Table 2.1. For the rest, leave the default values. Save your configuration under the name you wish (e.g. Brutus), since this will speed up this process in the future (if you don’t save it, you will have to type in everything each time you want to log in, instead of just loading it).

![PuTTY's configuration screen](image)

Figure 2.1: PuTTY’s configuration screen.

**TIP:** In the PuTTY configuration screen, expand the Window category from the left. Select Colours and in the Select a colour to adjust, choose Default Foreground. Change the Red, Green, and Blue values to 0, 255, and 0, respectively. Now take the red pill and log in (if you get this, you rock ☝!). You can customize many other parameters with any RGB color you wish. If you want, save your configuration.
Now just click *Open*. This will open a console. The console will be the way in which you will interact with Brutus. Enter your user name and press enter (→). It will then ask you for your password. Introduce it and type → again (when you type your password, you will not see the typical “*” nor the cursor will move: do not freak out, this is normal). If everything was done correctly, a welcome screen will be displayed and the console will be ready for your input (with something like [username@brutus2 ~]$).

If you are using other SSH client, see its corresponding documentation to learn how to connect to a remote system, although I guess the steps will be very similar to the ones presented here.

Congratulations! You have just logged to Brutus. However, you still need to know how to upload your files to Brutus so you can work with them. Let’s move on.

### 2.3 Getting an FTP client to manage files in Brutus

In order to move file into and out of Brutus, you will need a file transfer protocol (FTP) client. Personally, I recommend [WinSCP](https://winscp.net) but there are many others (like [Filezilla](https://filezilla-project.org)).

Go to [WinSCP’s download page](https://winscp.net) and choose either an installation package or a portable executable version, whichever you prefer. Once the download is finished, install it or move the files to the location you prefer (depending on what version you chose).

Launch WinSCP. You will open WinSCP’s login window (Fig. 2.2). Now just enter the parameters as shown in Table 2.1. For the rest, leave the default values. Again, save your configuration under the name you wish (e.g. Brutus), since this will speed up the login process for future sessions.

Now just click in *Login*. If by any chance a window pops up, just click *Yes*. If everything was done correctly, you will see WinSCP’s main screen, which is divided in two: on the left side, you will see the files that are your computer’s directories; on the right side you will see the files you have in your root directory in Brutus. Moving files between your computer and Brutus is as easy as dragging and dropping them as you wish. That’s it, no big deal.

**TIP:** When moving your files, remember *not* to execute multiple transfers at the same time. Doing so causes congestion and makes file transfer slower, not faster.

Once again, if you are using other FTP client, see its corresponding documentation to learn how to connect to a remote system and manage files,
although I guess the steps will be very similar to the ones presented here.

Way to go! Now you can move your files between your computer and Brutus at will. Eager to start running your heavy-computational-power-needed programs? Almost there.

2.4 Important Considerations

It is important to note that Brutus is protected by a firewall that allows only secure connections (SSH, SFTP, etc.) from within ETHZ’s internal network. If you want to connect from home, from another site, or from a public WiFi spot, you will need to use the [VPN service of ETHZ] to establish a secure connection to the ETHZ’s network first, or use another computer inside the ETH network as a gateway to the cluster (the latter is really not recommended if you are connecting from a public access point).

On the other hand, outgoing connections are unrestricted. You can therefore transfer files from the login nodes of Brutus to any computer in the world.
Table 2.1: Configuration Parameters for PuTTY and WinSCP

<table>
<thead>
<tr>
<th>Parameter</th>
<th>PuTTY</th>
<th>WinSCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>brutus.ethz.ch</td>
<td>brutus.ethz.ch</td>
</tr>
<tr>
<td>Port number</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Connection type</td>
<td>SSH</td>
<td>-</td>
</tr>
<tr>
<td>Close window on exit</td>
<td><em>Only on clean exit</em></td>
<td>-</td>
</tr>
<tr>
<td>File protocol</td>
<td>-</td>
<td><em>SFTP</em></td>
</tr>
</tbody>
</table>
Chapter 3

Using Brutus

3.1 Basics

You are now ready to start using Brutus (finally!). We will through the steps of launching a sample MATLAB application as an example. The same steps (more or less, give or take) are used for other types of applications.

Brutus is based in UNIX commands. Although there are a lot of workarounds to avoid using them (e.g. using WinSCP for file management tasks like creating directories), this is not geeky/cool enough for the purposes of this guide. I will try to explain step by step what commands to type. These should be more than enough for using Brutus. However, if you are interested, you may specific guides for that on the internet, like Rahardjo’s Summary of UNIX commands for example.

Before we continue, type in console

    $ bash ←

    With this, a record of the commands you input to the console will be registered. If you want to go through the commands you have used, just scroll with the up and down arrow keys. This will make life a little bit easier for you.

    It is a good practice to have a folder (directory) for each project. You can create one with the command mkdir. In this case, we will create the directory myGeekyDirectory, so you should type something like this:

    $ mkdir myGeekyDirectory ←

    How to know that the directory was actually created? The command ls will display all the current files and folders of your current directory.
$ ls ←
myGeekyDirectory

Now, we want to actually move into that directory, the equivalent of double clicking its icon. We can use the cd command (which, by the way, stands for “change directory”).

$ cd myGeekyDirectory ←

And the logical question now would be, how to know if we actually changed directory? The command pwd will display your current location.

$ pwd ←
/cluster/home/deparment/username/myGeekyDirectory

The next step would be to upload your files, in this case MATLAB scripts, to myGeekyDirectory. Since you already have your FTP client running this must be piece of cake. If you still don’t have an FTP client, immediately go to section [Getting an FTP client to manage files in Brutus](#) of Chapter 2 in page 6. Shame on you for skipping chapters while reading 🙄.

Let’s suppose you have a main.m script that calls three different functions, func1.m, func2.m, and func3.m. After uploading these files to myGeekyDirectory, the ls command will display

$ ls ←
main.m func1.m func2.m func3.m

Remember not to execute multiple transfers at the same time. Doing so causes congestion and makes file transfer slower, not faster.

In order to get the scripts going, we need to understand how Brutus works.

### 3.2 Modules

Brutus uses environment modules (or “modules” in short) as an elegant and user-friendly way of interacting with it. Modules allow you to load all the settings needed by a particular application and to unload them when you no longer need them.

To see what modules are available, type

$ module avail ←
This will display in console a long list of modules, including the available versions and which of them is the default one. It is important to note that the modules in the obsolete directory are no longer supported. They may or may not work. Therefore, they are not recommended.

In our case, we are interested in the MATLAB modules:

matlab/7.10(r2010a)
matlab/7.12(r2011a)
matlab/7.6(default:r2008a)
matlab/7.8(r2009a)
matlab/7.9(r2009b)

Strictly speaking, the name of each module is `matlab/7.10, matlab/7.12, etc. What is enclosed in parentheses is the version alias`. In other words, it is the same if you make reference to `matlab/7.10` or to `matlab/r2010a`. In this example, we will stick to the last convention, since it is more common to make reference to “MATLAB R2010a” rather than “MATLAB 7.10”.

To get a brief description of a module, type

$ module help matlab/r2010a

---------- Module Specific Help for 'matlab/7.10' ----------
MATLAB version 7.10 (x86_64)

To know what would a module exactly do, type

$ module show matlab/r2010a

-----------------------------------------------
/cluster/apps/modules/modulefiles/matlab/7.10:

module-whatis MATLAB version 7.10 (x86_64)
setenv MATLAB /cluster/apps/matlab/7.10
setenv MATLAB_BASEDIR /cluster/apps/matlab/7.10
prepend-path PATH /cluster/apps/matlab/7.10/bin
prepend-path LD_LIBRARY_PATH /cluster/apps/matlab/7.10/sys/os/glxa64
prepend-path LD_LIBRARY_PATH /cluster/apps/matlab/7.10/bin/glxa64
prepend-path LD_LIBRARY_PATH /cluster/apps/matlab/7.10/runtime/glxa64

However, knowing what modules are available and what they do is not enough. You need to `load` them. This is as easy as typing
$ module load matlab/r2010a

You can verify that the module was loaded correctly by typing

$ module list

Currently Loaded Modulefiles:
1) modules          2) matlab/7.10(r2010a)

The modules module is loaded by default. It is not strictly required - its main purpose is to allow you to access the manual pages of the module command.

In case you want to unload a module, you can use the command module unload. If you want to unload all the modules at once, use the command module purge. This will unload all currently loaded modules in LIFO style (last in, first out). Unloading modules in a different order may cause problems with some (context-sensitive) modules. If for some reason your environment is so corrupted that module purge is unable to clean up the mess, the best solution is to simply logout and login again. Keep in mind that modules affect only the shell in which the module command is executed and that they are unloaded every time you log out.

3.3 Submitting a (batch) job

Brutus is built as a batch system. In a batch system, a program takes a set of files as input, processes the data, and produces a set of files as an output. This operating mode is known as batch processing because the input data are collected into batches of files and processed in batches by the program. This particular characteristic allows Brutus to manage its resources between many different users and programs. It also allows it to administrate processing time between different jobs according to the availability of computing resources in an unsupervised way.

Submitting a job to the batch system is very easy. You actually need only one simple command: bsub. This is the command. Probably the most important command of all this document, so I think it is worth spending a few lines explaining it as good as possible.

The most basic way of using bsub with a third-party application (MATLAB, in this case) is as follows

$ bsub matlab -nojvm -singleCompThread -nodisplay -r scriptName.m
Which can be explained as follows:

- **bsub**
  The command itself. Stands for “batch submission”.

- **matlab**
  The third-party application with which you wish to run your script.

  - **-nojvm**
    This option should be used only when possible. If you don’t know when this might be the case, use it always. It disables the Java Virtual Machine.

  - **-singleCompThread**
    This option should be used only when possible. If you don’t know when this might be the case, use it always. It forces the execution to be done in a single computational thread.

  - **-nodisplay**
    Additional option to make execution more efficient.

  - **-r scriptName.m**
    Define what script will be executed.

However, this execution is very “stiff”. Brutus allows different options to make the execution as efficient and as suitable for your application as possible. A more complete submission would be

```
$ bsub -n N -W t -R "rusage[mem=MMMM]" matlab -nojvm -singleCompThread
-nodisplay -r scriptName.m ←
```

Which can be explained as follows:

- **bsub**
  The command itself. Stands for “batch submission”.

  - **-n N**
    Define the number N of multiple processors (threads) you require. By default, only 1 processor is assigned. Guest users (most users are guest users, very probably you are too) can use at most 128 processors, though this may change depending on the load of the cluster.
-w t
Define the time t your application will be ran. By default, only 1 hour of processing time is assigned. When the job reaches this limit, it is killed. The requested time can have the minutes format (e.g. 150 for 2.5 hours) or the HH:MM format (e.g. 02:30 for 2.5 hours).

-R "rusage[mem=MMMM]"
Define the amount of memory MMMM needed by your job (in MB) per processor. By default, 1024 MB (1 GB) of memory is assigned per processor.

- matlab
The third-party application application with which you wish to run your script.

-nojvm
This option should be used only when possible. If you don’t know when this might be the case, use it always. It disables the Java Virtual Machine.

-singleCompThread
This option should be used only when possible. If you don’t know when this might be the case, use it always. It forces the execution to be done in a single computational thread.

-nodisplay
Additional option to make execution more efficient.

-r scriptName.m
Define what script will be executed.

So, continuing with our humble example, the following submission

$ bsub -n 4 -W 720 -R "rusage[mem=4096]" matlab -nojvm
-singleCompThread -nodisplay -r main.m ←

would submit the script main.m in MATLAB in a single computational thread, with the Java Virtual Machine disabled, with 720 minutes (12 hours) assigned for execution time using 4 processors with 4096 MB each. Wow, that is a heavy computation. However, it is important to note that assigning a huge amount of resources to your submission does not guarantees that its execution will be faster, since this depends on your submission itself. For best performance, assign just a little bit more than the resources you expect your job will need.
After the submission, you will see on the console what ID number was assigned to your job (JobID) and in what queue will it be dispatched. You can check on the status of your job as follows

\[
\text{bjobs}
\]

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>481</td>
<td>username</td>
<td>PEND</td>
<td>pub.8h</td>
<td>brutus2</td>
<td>*y -r main</td>
<td>May 16</td>
<td>23:42</td>
</tr>
</tbody>
</table>

To kill one of your jobs type

\[
\text{bkkill \ JobID}
\]

To kill all of your jobs type

\[
\text{bkkill 0}
\]

### 3.4 Outputs

"Where are the outputs of my script?", you might be wondering. When a job you submitted is finished a file called `lsf.o<JobID>` will be created in your working directory (in this case, our beloved `myGeekyDirectory`). It is just a text file with information like when was the job completed, from where was it submitted, etc., as well as the program’s output and error messages that may have been sent to the standard output (in our MATLAB example, the standard output is MATLAB’s console).

If your script generates data that exceed 250 MB, you must request it when you submit the job (for example, `bsub -R "rusage[scratch=QQQQ]"`, where `QQQQ` is the space needed in MB).

**TIP**: Available space is always a concern, so remember to remove the output files, specially if they are heavy, as soon as possible. You wouldn’t like not having available scratch space because someone else has not removed his/her files, right?

### 3.5 Logging out

Once you are done, you may want to stop your connection to Brutus. This can easily be done with the commands **logout** or **exit**. The log out should not be affected nor should interfere with any jobs waiting in the queueing system or running on the compute nodes.

Congratulations! You have successfully used Brutus.
Chapter 4

Troubleshooting

Unfortunately, not everything goes according to planned every time. This chapter will try to give you some tips that may solve the most common problems you could potentially face while using Brutus. Please feel free to contribute with this list!

4.1 Important Considerations Using Different Software in Brutus

4.1.1 MATLAB

- Make sure you load the right MATLAB module (version). If your script uses a function that is not supported by the MATLAB version you loaded, you will get an error.

- Watch out for functions that are dependant on the OS or on previously installed software (e.g. \texttt{xlsread}).
Chapter 5

Closing comments

Hopefully, this guide helped you getting the basics using Brutus. Please note that what is shown here is far from exhaustive. What is presented here is the result of compiling different sources (full credit goes to them) and some hours of trial and error.

If you want to contribute in the expansion of this humble walkthrough with your personal experience using Brutus, if you want to report errors you may have found or if you have any suggestions, comments or (positive) criticism (©), please feel free to send me an email to arturomoncadatorres@gmail.com with “Brutus” in the subject.

5.1 Acknowledgements

I would like to thank Prof. Dr. Roger Gassert for giving me the opportunity to be part of a wonderful working team such as the Rehabilitation Engineering Laboratory (RELab).

5.2 Using this walkthrough

This walkthrough is originally hosted in www.arturomoncadatorres.com. However, feel free to use it and distribute it at your own will. Just please give proper credit (BibTex):


That’s all folks!
Bibliography


